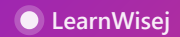


KI-Coding-Übung

Erstellen Sie das Beispielprojekt AccessOps dieses Moduls mit ChatGPT oder Claude aus einem fertigen Prompt, der das Modell auf die Spezifikation des Moduls verweist – und prüfen, starten und erweitern Sie dann, was Sie erhalten.



Der Assistent liefert die schnellen Hände. Der Architekt bleiben Sie.

Ein Assistent erstellt das Beispielprojekt dieses Moduls in wenigen Minuten. Was er nicht kann, ist zu entscheiden, ob das Ergebnis der richtige Entwurf für Ihre Anwendung ist – diese Entscheidung bleibt bei Ihnen. Das Urteilsvermögen bringen Sie mit: Instruieren Sie ihn sorgfältig und prüfen Sie dann, was zurückkommt, anhand dessen, was Ihnen Modul 2 gerade vermittelt hat.

Worum Sie bitten

Implementieren Sie die Authentifizierung von AccessOps: eine Login-Seite mit Sperre nach fünf Fehlversuchen, Passwortprüfung mit einem modernen gesalzenen Hash, einen SessionContext, der erst nach der Verifizierung erzeugt und in der Session abgelegt wird, ein Leerlauf-Timeout, das den Benutzer zum Login zurückschickt und den Zustand leert, einen expliziten Logout, der die Session beendet, und einen Audit-Eintrag für jeden Anmeldeversuch.

Das Modul verlangt diese Ergebnisse, und der Prompt unten fordert genau diese vom Assistenten an:

- ✓ Login-Seite mit Sperre und generischen Fehlermeldungen
- ✓ Service für Passwort-Hashing und -Verifizierung
- ✓ SessionContext mit verifizierter Identität
- ✓ Leerlauf-Timeout und expliziter Logout
- ✓ Audit-Einträge für Anmeldungen

Warum es so aufgebaut ist

AccessOps enthält Mitarbeiterdatensätze, vertrauliche Dokumente und Genehmigungsanträge – der erste Bildschirm, den jemand erreicht, ist also auch die erste Kontrolle, die zählt. Alles, was der Rest dieses Kurses durchsetzt – Berechtigungen, Regeln auf Datensatzebene, Audit-Trails –, beruht auf einer einzigen Annahme: dass der Server weiß, wer am anderen Ende dieser Session sitzt, und das belegen kann. Modul 1 hat die Vertrauensgrenzen gezogen; dieses Modul legt eine verifizierte Identität innerhalb der Session-Grenze ab und verhindert, dass sie erraten, ausgeliehen oder über ihre Lebensdauer hinaus genutzt wird.

Diese Begriffe muss der Assistent richtig verwenden. Geht die Antwort nachlässig mit ihnen um, hat er das Modul nicht verstanden:

Begriff	Was er hier bedeutet
gesalzener, langsamer Hash	Eine Einwegableitung des Passworts mit einem zufälligen Salt pro Benutzer und einer bewusst aufwendigen Iterationszahl, sodass eine gestohlene Tabelle nicht umgekehrt werden kann und jeder Rateversuch echte Rechenarbeit kostet.
Vergleich in konstanter Zeit	Zwei Hashes werden mit <code>CryptographicOperations.FixedTimeEquals</code> in einer festen Anzahl von Operationen verglichen, sodass die Antwortzeit des Servers nie verrät, wie viele Bytes eines Rateversuchs stimmten.
SessionContext	Das Objekt, das erst nach der Verifizierung eines Passworts erzeugt wird; es enthält Mitarbeiter-ID, Abteilung, Rollen und Anmeldedaten und liegt in <code>Application.Session</code> als der eine Ort, an dem jeder Code nachfragt, wer der Benutzer ist.
Session-Timeout	Die Leerlaufzeit, nach der Wisej.NET die serverseitige Session samt allem, was sie enthält, entsorgt – für die Anwendung in <code>Default.json</code> konfiguriert, statt pro Bildschirm geschätzt.
Session Fixation	Ein Angriff, bei dem der Bezeichner, der die anonyme Login-Seite getragen hat, anschließend auch die authentifizierte Session trägt, sodass ein vom Angreifer untergeschobener Bezeichner zu einem angemeldeten wird.
Login-Drosselung	Fehlversuche werden pro Konto und pro Client-Adresse gezählt und weitere Versuche für eine Sperrfrist abgewiesen – so wird aus einer unbegrenzten Ratemaschine eine Handvoll Versuche pro Stunde.
Login-Audit-Eintrag	Eine Zeile pro Versuch mit dem versuchten Benutzernamen, der Uhrzeit, der Client-Adresse und dem tatsächlichen Ergebnis – der Ort, an dem der konkrete Fehlergrund festgehalten wird, da der Bildschirm immer nur einen generischen anzeigt.

Selbst bauen – wählen Sie Ihren Assistenten

Beide Buttons öffnen einen neuen Tab, in dem **derselbe fertige Prompt** bereits eingetragen ist. Der Prompt verweist das Modell auf die Spezifikation dieses Moduls – eine einzelne Markdown-Datei mit der Lektion, der Lab-Aufgabe und den geforderten Ergebnissen –, sodass es mit dem echten Material arbeitet, statt zu raten, worum es in Modul 2 geht. Sie können [module-02.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden

Öffnet chatgpt.com mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.



Claude verwenden

Öffnet claude.ai mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie ihn stattdessen in einem leeren Ordner in ein Terminal einfügen.



Unverändert abgeschickt, erhalten Sie ein Projekt, kein Code-Fragment

Der Prompt verlangt am Ende die gesamte Projektmappe als einzelne `.zip`-Datei mit Anleitung zum Kompilieren und Starten. Laden Sie sie herunter, öffnen Sie die Projektmappe in Visual Studio, drücken Sie F5, und AccessOps startet auf `localhost`. Sie benötigen das .NET SDK und das NuGet-Paket `Wisej-4`; alles andere soll der Assistent laut Prompt selbst schreiben.

Ich arbeite gerade Modul 2 – „Authentifizierung und Session-Identität“ – des LearnWisej-Kurses „Sicherheits- und Authentifizierungsmuster“ durch und möchte das Beispielprojekt des Moduls gemeinsam mit dir erstellen. Bitte antworte auf Deutsch.

Lies zuerst die Modulspezifikation. Sie ist eine einzelne Markdown-Datei (auf Englisch) mit der vollständigen Lektion, der Lab-Aufgabe und den geforderten Ergebnissen:

<https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-02.md>

Erstelle dann ein vollständiges Wisej.NET-4-Projekt, das ich in Visual Studio öffnen kann:

1. Projekt „AccessOps“, C#, net8.0 oder neuer, NuGet-Paket Wisej-4 4.1.x, Kestrel-Host, der `app.UseWisej()` aufruft.
2. Setze um, was das Lab der Spezifikation verlangt: Implementieren Sie die Authentifizierung von AccessOps: eine Login-Seite mit Sperre nach fünf Fehlversuchen, Passwortprüfung mit einem modernen gesalzenen Hash, einen SessionContext, der erst nach der Verifizierung erzeugt und in der Session abgelegt wird, ein Leerlauf-Timeout, das den Benutzer zum Login zurückschickt und den Zustand leert, einen expliziten Logout, der die Session beendet, und einen Audit-Eintrag für jeden Anmeldeversuch.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Login-Seite mit Sperre und generischen Fehlermeldungen
 - Service für Passwort-Hashing und -Verifizierung
 - SessionContext mit verifizierter Identität
 - Leerlauf-Timeout und expliziter Logout
 - Audit-Einträge für Anmeldungen
4. Halte dich an die Spezifikation, wo immer sie eine Wisej.NET-Klasse, -Eigenschaft oder ein -Event nennt – verwende genau diese. Wo sie eine Wahl offenlässt, sag mir, welche API du gewählt hast und warum.
5. Der Server bleibt maßgeblich: Zustand und Validierung leben in .NET, der Browser stellt dar und meldet zurück. Verlagere nie eine Entscheidung nach JavaScript, nur weil es dort einfacher ist.
6. Mach das Ergebnis vorführbar: Ich muss es starten und das Verhalten sehen können, um das es in diesem Modul geht – auch, was passiert, wenn etwas schiefgeht.

Bevor du Code schreibst, fasse in eigenen Worten zusammen, was du bauen wirst und wo jedes Teil liegen wird, damit ich den Entwurf korrigieren kann, solange das noch wenig kostet.

Wenn der Code fertig ist, packe die gesamte Projektmappe in eine einzelne .zip-Datei zum Herunterladen und nenne mir die genauen Schritte, um sie zu kompilieren und auszuführen.

Beantworte zum Schluss diese Fragen aus der Spezifikation und nutze dabei deine eigene Implementierung als Beleg:

- Ihre Login-Seite beantwortet ein falsches Passwort in 480 ms und einen unbekannten Benutzernamen in 6 ms. Erklären Sie, was ein Angreifer allein aus diesem Unterschied erfährt, und beschreiben Sie die zwei Änderungen, die das Signal beseitigen, ohne die Meldung zu ändern.
- Ein Kollege schlägt vor, den angemeldeten Mitarbeiter in einer statischen Eigenschaft zu halten, weil „das einfacher ist, als jedes Mal in die Session zu greifen“. Beschreiben Sie genau, was der zweite gleichzeitige Benutzer sieht und in welchem Moment sich die beiden Sessions zum ersten Mal in die Quere kommen.
- Ihr Logout-Button leert den SessionContext, beendet die Session aber nicht. Nennen Sie zwei Dinge, die danach noch wiederherstellbar sind, und sagen Sie, welches Sie in einem Portal mit vertraulichen Dokumenten für das gravierendere halten.

Frag mich nach allem, was unklar ist, bevor du anfängst.

Prüfen Sie dann, was zurückkommt

Code, der kompiliert, ist noch lange kein richtiger Code. Die spannenden Fragen sind Designfragen: Ist das wirklich das Wisej.NET-Control, das die Aufgabe verlangt? Folgt der Code dem Lebenszyklus, den das Framework vorgibt? Liegt der maßgebliche Zustand auf dem Server – oder wurde er im Browser abgelegt, weil das bequemer war? Lesen Sie das Ergebnis als Reviewer, nicht als Kunde.

Die Checkliste liefert Ihnen dieses Modul. Beantworten Sie jeden Punkt anhand des Codes, den Sie gerade erhalten haben:

- ✓ Ihre Login-Seite beantwortet ein falsches Passwort in 480 ms und einen unbekannten Benutzernamen in 6 ms. Erklären Sie, was ein Angreifer allein aus diesem Unterschied erfährt, und beschreiben Sie die zwei Änderungen, die das Signal beseitigen, ohne die Meldung zu ändern.
- ✓ Ein Kollege schlägt vor, den angemeldeten Mitarbeiter in einer statischen Eigenschaft zu halten, weil „das einfacher ist, als jedes Mal in die Session zu greifen“. Beschreiben Sie genau, was der zweite gleichzeitige Benutzer sieht und in welchem Moment sich die beiden Sessions zum ersten Mal in die Quere kommen.
- ✓ Ihr Logout-Button leert den SessionContext, beendet die Session aber nicht. Nennen Sie zwei Dinge, die danach noch wiederherstellbar sind, und sagen Sie, welches Sie in einem Portal mit vertraulichen Dokumenten für das gravierendere halten.

Alles, was Sie korrigieren mussten, sollten Sie festhalten. Eine kurze Notiz zu den Konventionen und Entscheidungen, auf die Sie sich festgelegt haben – eingefügt in die erste Nachricht Ihrer nächsten Sitzung –, ist mehr wert als jeder noch so clevere Prompt.



Für diesen Kurs gibt es noch kein Referenzbeispiel – Ihres ist das erste

Das [LearnWisej-Beispiel-Repository](#) enthält ausgearbeitete Builds für die meisten Kurse, für diesen aber noch nicht. Es gibt also nichts zum Vergleichen und keine Abkürzung: Die Spezifikation und Ihr eigenes Review sind das ganze Sicherheitsnetz. Behalten Sie, was Sie bauen – und wenn es gut wird, würden wir es gern sehen.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen AccessOps-Code, als würdest du den Pull-Request eines Kollegen prüfen und hättest ihn nicht selbst geschrieben. Wo lebt der Zustand, was passiert bei einer einzelnen Benutzeraktion von Anfang bis Ende, was würdest du zur Überarbeitung zurückgeben, und welche Teile würdest du nicht guten Gewissens einem Kunden vorlegen? Sei konkret und zitiere die Zeilen, die du meinst – fasse mir nicht den Entwurf zusammen."*

- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses AccessOps-Projekts, der am offensichtlichsten nur ein Platzhalter für etwas Echtes ist – eine Fake-Datenquelle, ein In-Memory-Speicher, ein vereinfachter Service – und ersetze ihn durch eine praxisnahe Implementierung. Der Vertrag auf beiden Seiten darf sich nicht ändern: dieselben Typen, dieselben Aufrufe, dasselbe Verhalten im Fehlerfall. Sag mir, was du durch die echte Implementierung behandeln musstest, was der Platzhalter nicht behandelt hat."*
- ✓ **Machen Sie es fit für einen schlechten Tag.** *"Untersuche dieses AccessOps-Projekt auf Fehlerszenarien, die es noch nicht behandelt – fehlerhafte oder fehlende Eingaben, eine Abhängigkeit, die langsam oder nicht erreichbar ist, Arbeit, die eintrifft, nachdem der Benutzer die Seite verlassen hat, und zwei Aktionen, die sich gegenseitig in die Quere kommen. Sag mir für jedes zuerst, ob es in Wisej.NET 4 ein echtes Risiko ist und warum, und schreib erst dann die Korrektur. Wo du entscheidest, dass nichts behandelt werden muss, sag das und begründe es, statt defensiven Code hinzuzufügen."*



Nehmen Sie es mit ins Lab

Das praktische Lab am Ende dieses Moduls verlangt dieselben Ergebnisse, die Sie gerade vom Assistenten haben erstellen lassen. Nutzen Sie, was Sie hier haben, als Gerüst für Ihre Antwort und als Beleg dafür – nicht als die Antwort selbst. Bewertet wird im Lab Ihre Argumentation, und dieser Teil muss von Ihnen stammen.